AFRL-IF-RS-TR-2002-140
Final Technical Report
June 2002

TOPOLOGY BASED DOMAIN SEARCH (TBDS)

University of Southern California

Sponsored by
Defense Advanced Research Projects Agency
DARPA Order No. B550

APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

This report has been reviewed by the Air Force Research Laboratory, Information Directorate, Public Affairs Office (IFOIPA) and is releasable to the National Technical Information Service (NTIS). At NTIS it will be releasable to the general public, including foreign nations.

AFRL-IF-RS-TR-2002-140 has been reviewed and is approved for publication.

APPROVED:

GERALD B. LEPPERT
Project Engineer

FOR THE DIRECTOR:

WARREN H. DEBANY, Technical Advisor
Information Grid Division
Information Directorate

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. AGENCY USE ONLY (Leave blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	JUNE 2002	Final Jun 99 – Aug 01

4. TITLE AND SUBTITLE
TOPOLOGY BASED DOMAIN SEARCH (TBDS)

5. FUNDING NUMBERS
C - F30602-99-1-0523
PE - 62301E
PR - H550
TA - 10
WU - 01

6. AUTHOR(S)
William Manning

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
University of Southern California
Information Sciences Institute
4676 Admiralty Way, Suite 1001
Marina del Rey California 90292-6601

8. PERFORMING ORGANIZATION REPORT NUMBER
N/A

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Defense Advanced Research Projects Agency	AFRL/IFGA
3701 North Fairfax Drive	525 Brooks Road
Arlington Virginia 22203-1714	Rome New York 13441-4505

10. SPONSORING / MONITORING AGENCY REPORT NUMBER
AFRL-IF-RS-TR-2002-140

11. SUPPLEMENTARY NOTES
AFRL Project Engineer: Gerald B. Leppert/IFGA/(315) 330-1882/ Gerald.Leppert@rl.af.mil

12a. DISTRIBUTION / AVAILABILITY STATEMENT
APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

12b. DISTRIBUTION CODE

13. ABSTRACT ***(Maximum 200 Words)***
This effort will explore radical changes in the way Domain Name System (DNS) is used by endpoints in a network to improve the resilience of the endpoint and its applications in the face of dynamically changing infrastructure topology. This project will exploit the opportunity to remove the intrinsic dependence of a single, fixed root context for the DNS and develop a scalable architecture and localized optimization algorithms for constructing a dynamic, topologically sensitive root context for any network topology.

14. SUBJECT TERMS
Domain Name System, Dynamically Changing Infrastructure Topology

15. NUMBER OF PAGES
62

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	UNCLASSIFIED	UNCLASSIFIED	UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. Z39-18
298-102

Table of Contents

1. Introduction

The TBDS project developed extensions to existing network services to enable application client and server software to become more resilient to changes in topology by dynamically sensing changes and switching between client/server and peer-peer methods for both end system-to-server and server-to-server communications.

The first existing network service to be investigated was the Domain Name Systems (DNS) which is used to map symbolic Internet names to numeric Internet addresses. Based upon a hierarchical tree structure, the DNS relies upon uninterrupted connectivity of nodes to a special set of static, manually configured root servers. To improve the robustness and availability of the DNS service, TBDS developed and defined enhancements that enable nodes to map names to numbers without the need for uninterrupted connectivity to the Internet root servers. These techniques were automated, allowing transition between connected and unconnected operations to done without direct human intervention.

APPROACH

To accommodate the transient connectivity of individual nodes or entire sub networks, TBDS fabricated techniques to allow such nodes or networks to reconstruct the essential information that would be available to them if they had uninterrupted access to the global root servers (i.e. the root context).
In essence, a node in a disconnected partition is able to determine dynamically the root context for its network partition. Such modes of partitioned or disconnected operations are becoming increasingly important in the Internet, e.g. military operations, survivable networks, mobile nodes, and ad hoc wireless networks.

TBDS consists of software enhancements to the current DNS reference code base. The technical approach involved significant upgrades to the capabilities of the resolver module that resides in each client. Although the resolver will still function in client-server mode, it will also be able to function in purely peer-based relationship with other enhanced resolvers. Using multicast to discover other DNS resources, TBDS employed simple heuristics to construct a trusted, accurate root context for each node in a partition. The expected use of quorum voting was not realized due to the unstable nature of the development code base during the project duration. Given the potential for maliciously spoofed or faulty information about resources, TBDS exploited authentication services utilizing keys distributed from within the DNS. Authentication services were based on the DARPA-supported DNSSEC code base.

Evaluation of TBDS occurred in systematic phases. Extensive testing in an isolated laboratory preceded rollout and experimentation in selected research test bed networks, such as CAIRN. Our partners in testing included Microsoft, Apple, and Sun. We did not extend testing and evaluation beyond that phase.

2. TBDS Objectives

The software developed under the TBDS project included:

Enhancements to DNS server code based on the open source BIND to support reception and processing of multicast packets. Extensions to the initial configuration process, adding SRV records to the local host profile which are intended to bootstrap the process of DNS server identification.

We were unable to complete the software modules to allow creation and integration of authentication and policy evaluation modules in the standard DNS resolver libraries.

3. Accomplishments

YEAR: 1999

During the project, we completed the construction of a prototype development environment (the lab, upgrades, rewiring, upgrades etc.). This lab infrastructure became multicast capable and we began to test our multicast aware code. We upgraded four modules and two libraries in the BIND 8.1.2 release for supporting multicast. We also began porting these modifications to BIND 8.2.2 beta release to be in sync with DNSSEC capabilities that were not stable in the 8.1.2 release. We attended three working sessions with ISC staff to discuss the architecture of an enhanced resolver with a "goal engine". This type of capability was required for TBDS and was scheduled for alpha release in BIND in the second quarter of the year 2000. We met with Apple and Sun representatives to form a working group in the IETF and promoted TBDS as a component of "Zero Configuration Networking". We provided Apple and Sun with the modified code base and updated an IETF Internet draft describing the use of multicast for DNS service configuration with an IANA/ICANN assigned "well known multicast address". The assigned number is: a relative offset of 4 from the scoped multicast address. We participated in the CARIN DNSSEC workshop that was held on September 29-30 1999 at the ISI-East facilities in Washington, D.C., and published the Multicast Discovery Architecture document.

We designed and coded modifications to the BIND 8.1.2 DNS code base so it was multicast-aware. Skeleton architecture for authentication of DNS updates has been designed and was reviewed by ISC. Baseline conformance testing of the DNSSEC code base and supporting code was completed.

We completed an initial tradeoff analysis of methods for imprinting nodes with their own data and the bootstrap capabilities needed to query for other servers. Built a standardized PERL script that is able to configure Linux, Solaris, and FreeBSD machines. The configuration methodology required upgrading resolver modules to that found in ISC's BIND 8.2.2-patch5, to be able to support the SRV resource record. Extensive effort was spent with early releases of the BIND 9.x code, which was a complete rewrite of the mainline code and associated libraries. The majority of the time was spent working out the interactions with the different MTU sizes that support unfragmented UDP packets in the new libraries. These libraries have been rewritten to support eDNSzero and IPv6. We anticipated needing the larger packet sizes to transport authentication information.

With the active work being done in the IETF and elsewhere on policy expression languages, additional time was spent searching current literature that delayed the fleshing out of an authentication and policy evaluation framework. The work on policy algebra, by Brad Smith at UCSC looked like a promising candidate for defining our baseline methodology. Participation in the IETF Policy working group and the spatial location BOF provided more insight and understanding of requirements for disconnected operations.

A testing framework was used, based on the BIND feature interoperability bakeoffs. A more rigorous testing between software versions was needed to ensure cross platform interoperability. As identified earlier, the DNSSEC code base was not quite ready for use and it appeared that there were still outstanding issues regarding its viability. A possible data integrity problem was identified when forwarding is used. This concern was forwarded on to the ISC/Nominum staff. Our plan to execute a prototype evaluation engine was a more complex task than originally anticipated.

YEAR: 2000

By working with engineers from Apple Computer, we redefined the multicast search code to use a link-local multicast address: 224.0.0.251. This address was assigned by the IANA/ICANN. Use of link-local addressing added flexibility to the scoped multicast prefix that was already assigned. With link-local capability, there is an extra margin of safety in ensuring that the multicast requests are bounded.

Proof of concept modifications to BIND 8.1.2 were made to show multicast awareness could be added to BIND. An analysis was made of the existing DNS code deployment and the schedule of new feature deployment so that we could synchronize TBDS with a more appropriate code base. Testing identified a race condition due to overloading the semantics of the DNS Opcode that was used to communicate between servers.

This race condition was explored with ISC personnel in our use of existing DNS Opcodes. Discussion within the team and with others in the IETF led to the idea that we needed a new Opcode that would not overload the semantics of existing Opcodes. The original design specification presumes that few servers exist. To correct this problem a new Opcode was designed to disambiguate TBDS requests from normal nameserver requests. This new code will be added to the Bindv9 source tree. This condition, in conjunction with findings in the previous reporting period, led us to select ISC's BIND version 9 code as the appropriate base for future TBDS development.

Bindv9 was finally released and we began the process of porting the multicast packet generation/reception capabilities.

New tools were needed for instrumentation of packet tracing on wire, since there were few good multicast debugging tools. We modified ethereal, a packet tracing tool to support multicast. To provide an active, although restricted test bed, we began to maintain DNSSEC signed zones in the ip6.int tree. This tree was for the use in the 6bone, an IPv6 test network.

Working within the IETF, we coordinated DNS implementation Interoperability testing "bake-offs". DNS code from ISC, Cisco, Microsoft, Lucent and others were regular participants.

An internet draft of the new Opcode was prepared and circulated for comment. We took feedback from the Bind developers at ISC and within the IETF community. Additional questions were raised on the feasibility of depending on scoped multicast. Further investigations into other methods were explored.

We shifted our focus to compatibility and interoperability issues due to a delay in the release of the base code from ISC. In the past year, ISC has been joined by a variety of vendors who have been developing their own DNS code. It became apparent that the DNS specifications were not as crisp and well defined as they might be, even for baseline record processing.

An open invitation was made to known DNS developers by the Internet Software Consortium's Executive Director, David Conrad, the DNSEXT working group co-chair, Olafur Gundafson, and Bill Manning, to participate in what has become a series of workshops that test one or more features of the DNS specification. The TBDS project has been a participant nearly all of these sessions, participating in the workshops that have been held March 2000 in Adelaide, Australia, during the 47th IETF, August 2000 in Pittsburgh, PA, during the 48th IETF, and December 2000 in San Diego during the 49th IETF. We compiled a standardized test suite that can test conformance to the DNS specifications. Currently the suite only contains tests for seven resource record types, which is a small subset of the defined and used set.

One of the key components of TBDS depended on the successful deployment of components of the DNSsec suite of features. We participated in two DNSsec workshops. The first was held in conjunction with the North American Network Operators Group meeting in Washington D.C. in October. Verisign and NAI hosted this meeting with the focus on automated key creation and verification. The second was held in San Diego in December following the 49th IETF. This workshop was hosted by the WIDE project, EP.NET, LLC and Greenflash Consulting. This workshop focused on the interactions between a native IPv6 environment and DNSsec. The results of the workshop reinforced the demands on bandwidth that a TBDS aware environment places on the visible topologies. There were additional indications on the amount of processing capabilities needed to do per-query validation of the accompanying authentication material.

The DISCOVER Opcode Internet draft was written and submitted to the IETF secretary.

YEAR: 2001

We began transition work to have a more persistent organization take one the tasks associated with interoperability testing for DNS features. Meetings with the UltraDNS Corporation appeared favorable with transition scheduled prior to IETF-51. The DISCOVER Opcode was reviewed by Microsoft and Sun and was integrated into the BIND 9.3 code train. ISC and Nominum expected this code to be available in early 2002. This will complete the integration of multicast support into the reference implementation of DNS.

The integration of authentication and policy evaluation modules in the standard DNS resolver libraries proved problematic with the delays in a stable release of the open source BIND code from ISC. Major rework of these libraries was a scheduled component of their work for DISA. Unfortunately, this rework has been more time consuming than originally anticipated. Our dependence on the stability of this code did not allow us to complete this portion of the objectives.

We completed the documentation on the use of SRV resource records and the profile on how the resolver would evaluate multiple responses with different data. We did not have the personnel resources to complete the final objective.

The documentation on the second facet of multicast aware DNS, DISCOVER, was published as Internet Draft as is pending Experimental RFC status.

Interoperability testing of DNS implementations has been transitioned to UltraDNS as the lead party.

CURRENT PLAN

This project has ended and there are no further activities that have been funded by DARPA. Technology transition and standardization activities will continue to the extent that they can be supported by other funding sources.

TECHNOLOGY TRANSITION

Microsoft has picked up from the TBDS multicast work done and is embedding the basic functionality into its new products. An overview of their extensions may be found in: <ftp://ftp.isi.edu/in-notes/search.ietf.org/internet-drafts/draft-ietf-dnsext-mdns-1O.txt> (Appendix A). Apple is proposing modifications to the Dynamic Host Configuration Protocol (DHCP) to include techniques described in the TBDS work. This work may be found in the following drafts: <ftp://ftp.isi.edu/in-notes/search.ietf.org/internet-drafts/draft-cheshire-dnsext-multicastdns-OO.txt> (Appendix C) and <ftp://ftp.isi.edu/in-notes/search.ietf.org/internet-drafts/draft-cheshire-dnsext-nias-OO.txt> (Appendix B).

3.1 Publications

<ftp://ftp.isi.edu/in-notes/search.ietf.org/internet-drafts/draft-ymbk-opcode-discover-O3.txt> (Appendix D).

4. TBDS Staff

William Manning, Project Leader

Jeanine Yamazaki, Project Support

Alba Regalado, Project Support

Personnel Changes:

None

5. Project Timeline

No significant change.

6. Funding Requirements

No significant change.

7. Actual Accomplishments vs. Contract Deliverables

The project has concluded.

Appendix A
LinkLocal Multicast Name Resolution (LLMNR)

DNSEXT Working Group
INTERNET-DRAFT
Category: Standards Track
<draft-ietf-dnsext-mdns-10.txt>
23 March 2002

Levon Esibov
Bernard Aboba
Dave Thaler
Microsoft

Linklocal Multicast Name Resolution (LLMNR)

Abstract

Today, with the rise of home networking, there are an increasing number of ad-hoc networks operating without a DNS server. In order to allow name resolution in such environments, Link-Local Multicast Name Resolution (LLMNR) is proposed.

1. Introduction

This document discusses Link-Local Multicast Name Resolution (LLMNR), which operates on a separate port from DNS, with a distinct resolver cache, but does not change the format of DNS packets.

The goal of LLMNR is to enable name resolution in scenarios in which conventional DNS name resolution is not possible. These include scenarios in which hosts are not configured with the address of a DNS server.

Since IPv4 and IPv6 utilize distinct configuration mechanisms, it is possible for a dual stack host to be configured with the address of a DNS server for IPv4, while remaining unconfigured with a DNS server suitable for use with IPv6. Since automatic IPv6 DNS configuration mechanisms such as [DHCPv6DNS] and [DNSDisc] are not yet widely deployed, such "partially configuration" may be common in the short term. However, in the long term, IPv6 DNS configuration will become more common so that LLMNR will typically be restricted to adhoc networks in which neither IPv4 nor IPv6 DNS servers are configured.

Service discovery in general, as well as discovery of DNS servers using LLMNR in particular is outside of the scope of this document, as is name resolution over non-multicast capable media.

In this document, the key words "MAY", "MUST, "MUST NOT", "OPTIONAL", "RECOMMENDED", "SHOULD", and "SHOULD NOT", are to be interpreted as described in [RFC2119].

2. Name resolution using LLMNR

While operating on a different port with a distinct resolver cache, LLMNR makes no change to the current format of DNS packets.

LLMNR queries are sent to and received on port 5353 using a LINKLOCAL address as specified in "Administratively Scoped IP Multicast" [RFC2365] for IPv4 and the "solicited name" LINKLOCAL multicast addresses for IPv6, and using a unicast addresses in a few scenarios described below in Section 3. The LLMNR LINKLOCAL address to be used for IPv4 is 224.0.0.251. LINKLOCAL addresses are used to prevent propagation of LLMNR traffic across routers, potentially flooding the network.

Propagation of LLMNR packets on the local link is considered sufficient to enable name resolution in small networks. The assumption is that if a network has a home gateway, then the network either has a DNS server or the home gateway can function as a DNS proxy. By implementing DHCPv4 as well as a DNS proxy and dynamic DNS, home gateways can provide name resolution for the names of IPv4 hosts on the local network.

For small IPv6 networks, equivalent functionality can be provided by a home gateway implementing DHCPv6 for DNS configuration [DHCPv6DNS], as well as a DNS proxy supporting AAAA RRs and dynamic DNS, providing name resolution for the names of IPv6 hosts on the local network.

This should be adequate as long as home gateways implementing DNS configuration also support dynamic DNS in some form. If the home gateway only supports DNS discovery [DNSDisc] but not DHCPv6 DNS configuration [DHCPv6DNS] or dynamic client update, then resolution of the names of IPv6 hosts on the local link will not be possible. Since IPv6 DNS discovery will configure the DNS server address, LLMNR will not be enabled by default. Yet without gateway support for client dynamic update or DHCPv6, dynamic DNS will not be enabled.

In the future, LLMNR may be defined to support greater than LINKLOCAL multicast scope. This would occur if LLMNR deployment is successful, the assumption that LLMNR is not needed on multiple links proves incorrect, and multicast routing becomes ubiquitous. For example, it is not clear that this assumption will be valid in large adhoc networking scenarios.

Once we have experience in LLMNR deployment in terms of administrative issues, usability and impact on the network it will be possible reevaluate which multicast scopes are appropriate for use with multicast name resolution mechanisms.

2.1. Behavior of the sender and responder

For the purpose of this document a host that sends a LLMNR query is called a "sender", while a host that listens to (but not necessarily responds to) a LLMNR query is called "responder". Although the same host may be configured as a "sender", but not a "responder" and vice versa, i.e. as a "responder", but not a "sender", the host configured as a "responder" MUST act as a sender by using LLMNR dynamic update requests to verify the uniqueness of names as described in Section 5.

2.1.1. Behavior of senders

A sender sends an LLMNR query for any legal Type of resource record (e.g. A, PTR, etc.) to the LINKLOCAL address. Notice that in some scenarios described below in Section 3 a sender may also send a unicast query. The RD (Recursion Desired) bit MUST NOT be set. If a responder receives a query with the header containing RD set bit, the responder MUST ignore the RD bit.

The IPv6 LINKLOCAL address a given responder listens to, and to which a sender sends, is a link-local multicast address formed as follows: The name of the resource record in question is expressed in its canonical form (see [RFC2535], section 8.1), which is uncompressed with all alphabetic characters in lower case. The first label of the resource record name is then hashed using the MD5 algorithm, described in [RFC1321]. The first 32 bits of the resultant 128-bit hash is then appended to the prefix FF02:0:0:0:0:2::/96 to yield the 128-bit "solicited name multicast address". (Note: this procedure is intended to be the same as that specified in section 3 of "IPv6 Node Information Queries" [NodeInfo]). A responder that listens for queries for multiple names will necessarily listen to multiple of these solicited name multicast addresses.

If the LLMNR query is not resolved during a limited amount of time (LLMNR_TIMEOUT), then a sender MAY repeat the transmission of a query in order to assure themselves that the query has been received by a host capable of responding to the query. The default value for LLMNR_TIMEOUT is 1 second.

Repetition MUST NOT be attempted more than 3 times and SHOULD NOT be repeated more often than once per second to reduce unnecessary network traffic. The delay between attempts should be randomized so as to avoid synchronization effects.

2.1.2. Behavior of responders

A responder listens on port 5353 on the LINKLOCAL address and on the unicast address(es) that could be set as the source address(es) when the responder responds to the LLMNR query. Responders MUST respond to LLMNR queries to those and only those names for which they are authoritative. As an example, computer "host.example.com." is authoritative for the domain "host.example.com.". On receiving a LLMNR A record query for the name "host.example.com." such a host responds with A record(s) that contain IP address(es) in the RDATA of the record.

In conventional DNS terminology a DNS server authoritative for a zone is authoritative for all the domain names under the zone root except for the branches delegated into separate zones. Contrary to conventional DNS terminology, a responder is authoritative only for the zone root. For example the host "host.example.com." is not authoritative for the name "child.host.example.com." unless the host is configured with multiple names, including "host.example.com." and "child.host.example.com.". The purpose of limiting the name authority scope of a responder is to prevent complications that could be caused by coexistence of two or more hosts with the names representing child and parent (or grandparent) nodes in the DNS tree, for example, "host.example.com." and "child.host.example.com.".

In this example (unless this limitation is introduced) a LLMNR query for an A record for the name "child.host.example.com." would result in two authoritative responses: name error received from "host.example.com.", and a requested A record - from "child.host.example.com.". To prevent this ambiguity, LLMNR enabled hosts could perform a dynamic update of the parent (or grandparent) zone with a delegation to a child zone. In this example a host "child.host.example.com." would send a dynamic update for the NS and glue A record to "host.example.com.", but this approach significantly complicates implementation of LLMNR and would not be acceptable for lightweight hosts.

A response to a LLMNR query is composed in exactly the same manner as a response to the unicast DNS query as specified in [RFC1035]. Responders MUST never respond using cached data, and the AA (Authoritative Answer) bit MUST be set. The response is sent to the sender via unicast. A response to an LLMNR query MUST have RCODE set to zero. Responses with RCODE set to zero are referred to in this document as "positively resolved". LLMNR responders may respond only to queries which they can resolve positively.

If a TC (truncation) bit is set in the response, then the sender MAY use the response if it contains all necessary information, or the sender MAY discard the response and resend the query over TCP or using EDNS0 with larger window using the unicast address of the responder. The RA (Recursion Available) bit in the header of the response MUST NOT be set. Even if the RA bit is set in the response header, the sender MUST ignore it.

2.1.3. LLMNR addressing

For IPv4 LINKLOCAL addressing, section 2.4 of "Dynamic Configuration of IPv4 Link-Local Addresses" [IPV4Link] lays out the rules with respect to source address selection, TTL settings, and acceptable source/destination address combinations. IPv6 is described in [RFC2460]; IPv6 LINKLOCAL addressing is described in [RFC2373]. LLMNR queries and responses MUST obey the rules laid out in these documents.

In composing an LLMNR response, the responder MUST set the Hop Limit field in the IPv6 header and the TTL field in IPv4 header of the LLMNR response to 255. The sender MUST verify that the Hop Limit field in IPv6 header and TTL field in IPv4 header of each response to the LLMNR query is set to 255. If it is not, then sender MUST ignore the response.

Implementation note:

In the sockets API for IPv4, the IP_TTL and IP_MULTICAST_TTL socket options are used to specify the TTL of outgoing unicast and multicast packets. The IP_RECVTTL socket option is available on some platforms to receive the IPv4 TTL of received packets with recvmsg(). [RFC2292] specifies similar options for specifying and receiving the IPv6 Hop Limit.

2.1.4. Use of LLMNR TTL

The responder should use a pre-configured TTL value in the records returned in the LLMNR query response. Due to the TTL minimalization necessary when caching an RRset, all TTLs in an RRset MUST be set to the

same value. In the additional and authority section of the response the responder includes the same records as a DNS server would insert in the response to the unicast DNS query.

2.1.5. No/multiple responses

The sender MUST anticipate receiving no replies to some LLMNR queries, in the event that no responders are available within the linklocal multicast scope, or in the event that no positive non-null responses exist for the transmitted query. If no positive response is received, a resolver treats it as a response that no records of the specified type and class for the specified name exist (NXRRSET).

The sender MUST anticipate receiving multiple replies to the same LLMNR query, in the event that several LLMNR enabled computers receive the query and respond with valid answers. When this occurs, the responses MAY first be concatenated, and then treated in the same manner that multiple RRs received from the same DNS server would, ordinarily. However, after receiving an initial response, the sender is not required to wait for LLMNR_TIMEOUT for additional responses.

3. Usage model

The same host may be configured as a "sender", but not a "responder" and vice versa (as a "responder", but not "sender"). However, the host configured as a "responder" MUST at least use "sender’s" capability to send LLMNR dynamic update requests to verify the uniqueness of the names as described in Section 5. An LLMNR "sender" MAY multicast requests for any name. If that name is not qualified and does not end in a trailing dot, for the purposes of LLMNR, the implicit search order is as follows:

[1] Request the name with the current domain appended.
[2] Request just the name.

This is the behavior suggested by [RFC1536]. LLMNR uses this technique to resolve unqualified host names.

If a DNS server is running on a host that supports LLMNR, the DNS server MUST respond to LLMNR queries only for the RRSets owned by the host on which the server is running, but MUST NOT respond for the records for which the server is authoritative.

A sender MUST NOT send a unicast LLMNR query except when:

a. A sender repeats a query after it received a response to the previous LLMNR query with the TC bit set, or

b. The sender's LLMNR cache contains an NS resource record that enables the sender to send a query directly to the hosts authoritative for the name in the query.

A responder with a name "host.example.com." configured to respond to the LLMNR queries is authoritative for the name "host.example.com.". For example, when a responder with the name "host.example.com." receives an A type LLMNR query for the name "host.example.com." it authoritatively responds to the query.

The same host MAY use LLMNR queries for the resolution of the local names, and conventional DNS queries for resolution of other DNS names.

3.1. LLMNR configuration

LLMNR usage can be configured manually or automatically. On interfaces where no manual or automatic DNS configuration has been performed for a given protocol (IPv4 or IPv6), LLMNR SHOULD be enabled for that protocol.

For IPv6, the stateless DNS discovery mechanisms described in "IPv6 Stateless DNS Discovery" [DNSDisc] or "Using DHCPv6 for DNS Configuration in Hosts" [DHCPv6DNS] can be used to discover whether LLMNR should be enabled or disabled on a per-interface basis.

Where DHCPv4 or DHCPv6 is implemented, DHCP options can be used to configure LLMNR on an interface. The LLMNR Enable Option, described in [LLMNREnable], can be used to explicitly enable or disable use of LLMNR on an interface. The LLMNR Enable Option does not determine whether or in which order DNS itself is used for name resolution. The order in which various name resolution mechanisms should be used can be specified using the Name Service Search Option for DHCP, [RFC2937].

Note that it is possible for LLMNR to be enabled for use with IPv6 at the same time it is disabled for IPv4, and vice versa. For example, a home gateway may implement a DNS proxy and DHCPv4, but not DHCPv6 for DNS configuration [DHCPv6DNS] or stateless DNS discovery [DNSDisc]. In such a circumstance, IPv6 hosts will not be configured with a DNS server. Where DHCPv6 is not supported, it will not be possible for the DNS proxy within the home gateway to dynamically register names learned via DHCPv6. As a result, unless the DNS proxy supports client update, it will not be able to respond to AAAA RR queries for local names sent over IPv4 or IPv6, preventing IPv6 hosts from resolving the names of other

IPv6 hosts on the local link. In this situation, LLMNR enables resolution of dynamic names, and it will be enabled for use with IPv6, even though it is disabled for use with IPv4.

4. Sequence of events

The sequence of events for LLMNR usage is as follows:

1. If a sender needs to resolve a query for a name "host.example.com", then it sends a LLMNR query to the LINKLOCAL multicast address.

2. A responder responds to this query only if it is authoritative for the domain name "host.example.com". The responder sends a response to the sender via unicast over UDP.

3. Upon the reception of the response, the sender verifies that the Hop

   Limit field in IPv6 header or TTL field in IPv4 header (depending on the protocol used) of the response is set to 255. The sender then verifies compliance with the addressing requirements for IPv4, described in [IPV4Link], and IPv6, described in [RFC2373]. If these conditions are met, then the sender uses and caches the returned response. If not, then the sender ignores the response and continues waiting for the response.

5. Conflict resolution

There are some scenarios when multiple responders MAY respond to the same query. There are other scenarios when only one responder may respond to a query. Resource records for which the latter queries are submitted are referred as UNIQUE throughout this document. The uniqueness of a resource record depends on a nature of the name in the query and type of the query. For example it is expected that:

- multiple hosts may respond to a query for a SRV type record
- multiple hosts may respond to a query for an A type record for a cluster name (assigned to multiple hosts in the cluster)
- only a single host may respond to a query for an A type record for a hostname.

Every responder that responds to a LLMNR query and/or dynamic update request AND includes a UNIQUE record in the response:

1. MUST verify that there is no other host within the scope of the LLMNR query propagation that can return a resource record for the same name, type and class.
2. MUST NOT include a UNIQUE resource record in the response without having verified its uniqueness.

Where a host is configured to respond to LLMNR queries on more than one interface, the host MUST verify resource record uniqueness on each interface for each UNIQUE resource record that could be used on that interface. To accomplish this, the host MUST send a dynamic LLMNR update request for each new UNIQUE resource record. Format of the dynamic LLMNR update request is identical to the format of the dynamic DNS update request specified in [RFC2136]. Uniqueness verification is carried out when the host:

- starts up or
- is configured to respond to the LLMNR queries on some interface or
- is configured to respond to the LLMNR queries using additional UNIQUE resource records.

Below we describe the data to be specified in the dynamic update request:

Header section
Contains values according to [RFC2136].

Zone section
The zone name in the zone section MUST be set to the name of the UNIQUE record. The zone type in the zone section MUST be set to SOA. The zone class in the zone section MUST be set to the class of the UNIQUE record.

Prerequisite section
This section MUST contain a record set whose semantics are described in [RFC2136], Section 2.4.3 "RRset Does Not Exist", requesting that RRs with the NAME and TYPE of the UNIQUE record do not exist.

Update section
This section MUST be left empty.

Additional section
This section is set according to [RFC2136].

When a host that owns a UNIQUE record receives a dynamic update request that requests that the UNIQUE resource record set does not exist, the host MUST respond via unicast with the YXRRSET error, according to the rules described in Section 3 of [RFC2136].

After the client receives an YXRRSET response to its dynamic update request stating that a UNIQUE resource record does not exist, the host MUST check whether the response arrived on another interface. If this is the case, then the client can use the UNIQUE resource record in response to LLMNR queries and dynamic update requests. If not, then it MUST NOT use the UNIQUE resource record in response to LLMNR queries and dynamic update requests.

Note that this name conflict detection mechanism doesn't prevent name conflicts when previously partitioned segments are connected by a bridge. In such a situation, name conflicts are detected when a sender receives more than one response to its LLMNR query. In this case, the sender sends the first response that it received to all responders that responded to this query except the first one, using unicast. A host that receives a query response containing a UNIQUE resource record that it owns, even if it didn't send such a query, MUST verify that no other host within the LLMNR scope is authoritative for the same name, using the dynamic LLMNR update request mechanism described above.

Based on the result, the host detects whether there is a name conflict and acts as described above.

5.1. Considerations for Multiple Interfaces

A multi-homed host may elect to configure LLMNR on only one of its active interfaces. In many situations this will be adequate. However, should a host wish to configure LLMNR on more than one of its active interfaces, there are some additional precautions it MUST take. Implementers who are not planning to support LLMNR on multiple interfaces simultaneously may skip this section.

A multi-homed host checks the uniqueness of UNIQUE records as described in Section 5. The situation is illustrated in figure 1 below:

```
     ----------  ----------
      |      |    |      |
     [A]    [myhost]   [myhost]
```

Figure 1. LINKLOCAL name conflict

In this situation, the multi-homed myhost will probe for, and defend, its host name on both interfaces. A conflict will be detected on one interface, but not the other. The multi-homed myhost will not be able to respond with a host RR for "myhost" on the interface on the right (see Figure 1). The multi-homed host may, however, be configured to use the "myhost" name on the interface on the left.

Since names are only unique per-link, hosts on different links could be using the same name. If an LLMNR client sends requests over multiple interfaces, and receives replies from more than one, the result returned to the client is defined by the implementation. The situation is illustrated in figure 2 below.

```
     ----------  ----------
      |      |    |     |
     [A]    [myhost]   [A]
```

Figure 2. Off-segment name conflict

If host myhost is configured to use LLMNR on both interfaces, it will send LLMNR queries on both interfaces. When host myhost sends a query for the host RR for name "A" it will receive a response from hosts on both interfaces.

Host myhost will then forward a response from the first responder to the second responder, who will attempt to verify the uniqueness of host RR for its name, but will not discover a conflict, since the conflicting host resides on a different link. Therefore it will continue using its name.

Indeed, host myhost cannot distinguish between the situation shown in Figure 2, and that shown in Figure 3 where no conflict exists:

```
             [A]
            |   |
        -----   -----
            |   |
          [myhost]
```

Figure 3. Multiple paths to same host

This illustrates that the proposed name conflict resolution mechanism does not support detection or resolution of conflicts between hosts on different links. This problem can also occur with unicast DNS when a multi-homed host is connected to two different networks with separated name spaces. It is not the intent of this document to address the issue of uniqueness of names within DNS.

5.2. API issues

[RFC2553] provides an API which can partially solve the name ambiguity problem for applications written to use this API, since the sockaddr_in6 structure exposes the scope within which each scoped address exists, and this structure can be used for both IPv4 (using v4-mapped IPv6 addresses) and IPv6 addresses.

Following the example in Figure 2, an application on 'myhost' issues the request getaddrinfo("A", ...) with ai_family=AF_INET6 and ai_flags=AI_ALL|AI_V4MAPPED. LLMNR requests will be sent from both interfaces and the resolver library will return a list containing multiple addrinfo structures, each with an associated sockaddr_in6 structure. This list will thus contain the IPv4 and IPv6 addresses of both hosts responding to the name 'A'. Link-local addresses will have a sin6_scope_id value that disambiguates which interface is used to reach the address. Of course, to the application, Figures 2 and 3 are still indistinguishable, but this API allows the application to communicate successfully with any address in the list.

6. Security Considerations

LLMNR is by nature a peer to peer name resolution protocol, for use in situations when a DNS server is not configured. It is therefore inherently more vulnerable than DNS, since existing DNS security mechanisms are difficult to apply to LLMNR and an attacker only needs to be misconfigured to answer an LLMNR query with incorrect information.

In order to address the security vulnerabilities, the following mechanisms are contemplated:

[1] Scope restrictions.

[2] Usage restrictions.

[3] Cache and port separation.

[4] Authentication.

These techniques are described in the following sections.

6.1. Scope restriction

With LLMNR it is possible that hosts will allocate conflicting names for a period of time, or that attackers will attempt to deny service to other hosts by allocating the same name. Such attacks also allow hosts to receive packets destined for other hosts.

In the absence of authentication, LLMNR reduces the exposure to such threats by ignoring LLMNR query response packets received from off-link senders. In all received responses, the Hop Limit field in IPv6 and the TTL field in IPv4 are verified to contain 255, the maximum legal value. Since routers decrement the Hop Limit on all packets they forward, received packets containing a Hop Limit of 255 must have originated from a neighbor.

While restricting ignoring packets received from off-link senders reduces the level of vulnerability, it does not eliminate it. There are scenarios such as public "hotspots" where attackers can be present on the same link. These threats are most serious in wireless networks such as 802.11, since attackers on a wired network will require physical access to the home network, while wireless attackers may reside outside the home. Link-layer security can be of assistance against these threats if it is available.

6.2. Usage restriction

As noted in Section 3.1, LLMNR is intended for usage in scenarios where a DNS server is not configured. If an interface has been configured for a given protocol via any automatic configuration mechanism which is able to supply DNS configuration information, then LLMNR SHOULD NOT be used on that interface for that protocol unless it has been explicitly enabled, whether via that mechanism or any other. This ensures that upgraded hosts do not change their default behavior, without requiring the source of the configuration information to be simultaneously updated. This implies that on the interface, the host will neither listen on the LINKLOCAL multicast address, nor will it send queries to that address.

Violation of this guideline can significantly increases security vulnerabilities. For example, if an LLMNR query were to be sent whenever a DNS server did not respond in a timely way, then an attacker could execute a denial of service attack on the DNS server(s) and then poison the LLMNR cache by responding to the resulting LLMNR queries with incorrect information.

The vulnerability would be even greater if LLMNR is given higher priority than DNS among the enabled name resolution mechanisms. In such a configuration, a denial of service attack on the DNS server would not be necessary in order to poison the LLMNR cache, since LLMNR queries would be sent even when the DNS server is available. In addition, the LLMNR cache, once poisoned, would take precedence over the DNS cache, eliminating the benefits of cache separation.

As a result, LLMNR is best thought of as a name resolution mechanism of last resort, useful only in situations where a DNS server is not configured. Where resilience against DNS server failure is desired, configuration of additional DNS servers or DNS server clustering is recommended; LLMNR is not an appropriate "failsafe" mechanism.

6.3. Cache and port separation

In order to prevent responses to LLMNR queries from polluting the DNS cache, LLMNR implementations MUST use a distinct, isolated cache for LLMNR. The use of separate caches is most effective when LLMNR is used as a name resolution mechanism of last resort, since the this minimizes the opportunities for poisoning the LLMNR cache, and decreases reliance on it.

LLMNR operates on a separate port (5353) from DNS, reducing the likelihood that a DNS server will unintentionally respond to an LLMNR query.

6.4. Authentication

LLMNR does not require use of DNSSEC, and as a result, responses to LLMNR queries MAY NOT be authenticated. If authentication is desired, and a pre-arranged security configuration is possible, then IPsec ESP with a null-transform MAY be used to authenticate LLMNR responses. In a small network without a certificate authority, this can be most easily accomplished through configuration of a group pre-shared key for trusted hosts.

7. IANA Considerations

This specification does not create any new name spaces for IANA administration. Since it uses a port (5353) and link scope multicast IPv4 address (224.0.0.251) previously allocated for use with LLMNR, no additional IANA allocations are required.

8. Normative References

[RFC1035] Mockapetris, P., "Domain Names - Implementation and Specification", RFC 1035, November 1987.

[RFC1321] Rivest, R., "The MD5 Message-Digest Algorithm", RFC 1321, April 1992.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2136] Vixie, P., Thomson, S., Rekhter, Y., Bound, J., "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.

[RFC2365] Meyer, D., "Administratively Scoped IP Multicast", BCP 23, RFC 2365, July 1998.

[RFC2373] Hinden, R., Deering, S., "IP Version 6 Addressing Architecture", RFC 2373, July 1998.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC2535] Eastlake, D., "Domain Name System Security Extensions", RFC 2535, March 1999.

[IPV4Link] Cheshire, S., Aboba, B., "Dynamic Configuration of IPv4 Link-Local Addresses", Internet draft (work in progress), draft-ietf-zeroconf-ipv4-linklocal-05.txt, November 2001.

[LLMNREnable] Guttman, E., "DHCP LLMNR Enable Option", Internet draft (work in progress), draft-guttman-mdns-enable-02.txt, April 2002.

9. Informative References

[RFC1536] Kumar, A., et. al. "DNS Implementation Errors and Suggested Fixes", RFC 1536, October 1993.

[RFC2292] Stevens, W., Thomas, M., "Advanced Sockets API for IPv6", RFC 2292, February 1998.

[RFC2434] Alvestrand, H. and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

[RFC2553] Gilligan, R., Thomson, S., Bound, J., Stevens, W., "Basic Socket Interface Extensions for IPv6", RFC 2553, March 1999.

[RFC2937] Smith, C., "The Name Service Search Option for DHCP", RFC 2937, September 2000.

[DHCPv6DNS] Droms, R., Narten, T., and Aboba, B. "Using DHCPv6 for DNS Configuration in Hosts", draft-droms-dnsconfig-dhcpv6-01.txt, Internet draft (work in progress), March 2002.

[DNSDisc] Thaler, D., Hagino, I., "IPv6 Stateless DNS Discovery", Internet draft (work in progress), draft-ietf-ipngwg-dns-discovery-03.txt, November 2001.

[NodeInfo] Crawford, Matt, "IPv6 Node Information Queries", Internet draft (work in progress), draft-ietf-ipn-gwg-icmp-name-lookups-08.txt, July 2001.

Acknowledgments

This work builds upon original work done on multicast DNS by Bill Manning and Bill Woodcock. Bill Manning's work was funded under DARPA grant #F30602-99-1-0523. The authors gratefully acknowledge their contribution to the current specification. Constructive input has also been received from Mark Andrews, Stuart Cheshire, Robert Elz, Rob Austein, James Gilroy, Olafur Gudmundsson, Erik Guttman, Myron Hattig, Thomas Narten, Erik Nordmark, Sander Van-Valkenburg, Tomohide Nagashima and Brian Zill.

Authors' Addresses

Levon Esibov
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

EMail: levone@microsoft.com

Bernard Aboba
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Phone: +1 425 706 6605
EMail: bernarda@microsoft.com

Dave Thaler
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Phone: +1 425 703 8835

EMail: dthaler@microsoft.com

Intellectual Property Statement

Full Copyright Statement

Expiration Date

This memo is filed as <draft-ietf-dnsext-mdns-10.txt>, and expires October 22, 2002.

Appendix B
Discovering Named Instances of Abstract Services using DNS

Stuart Cheshire
Apple Computer

Document: draft-cheshire-dnsext-nias-00.txt
Expires 13th January 2002
13th July 2001

Discovering Named Instances of Abstract Services using DNS

<draft-cheshire-dnsext-nias-00.txt>

Status of this Memo

This document is an Internet-Draft and is in full conformance with all provisions of Section 10 of RFC2026. Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html

Distribution of this memo is unlimited.

Abstract

This document proposes a convention for naming and structuring DNS resource records that allows clients to discover a list of named instances of a particular given desired type of service.

1. Acknowledgements

This concepts described in this draft have been explored and developed with help from Bill Woodcock, Erik Guttman, and others.

2. Introduction

This is a rough first draft. Its purpose is to describe the proposed idea well enough for meaningful discussion to take place. As such, while feedback concerning typographical mistakes and similar minutiae is always appreciated, the reader is advised that it is probably unwise to waste a lot of time on such trivia until after we find out whether this proposal will even live long enough to become a 'draft-01'.

This document proposes a convention for naming and structuring DNS resource records that allows clients to discover a list of named instances of a particular given desired type of service.

This document proposes no change to the structure of DNS messages, and no new operation codes, response codes, resource record types, or any other new DNS protocol values. This document simply proposes a convention for how existing resource record types can be named and structured to facilitate service discovery.

This proposal is entirely compatible with today's existing unicast DNS server and client software.

This proposal is also compatible with the proposal for Multicast DNS outlined in "Performing DNS queries via IP Multicast" [mDNS-SC].

3. Design Goals

A good service discovery protocol needs to have three properties:

(i) The ability to query for services of a certain type in a certain logical domain and receive in response a list of named instances (network browsing, or "Service Instance Enumeration").

(ii) Given a particular named instance, the ability to efficiently resolve that instance name to the required information a client needs to actually use the service, i.e. IP address and port number, at the very least (Service Name Resolution).

(iii) Instance names should be relatively persistent. If a user selects their default printer from a list of available choices today, then tomorrow they should still be able to print on that printer -- even if the IP address and/or port number where the service resides have changed -- without the user (or their software) having to repeat the network browsing step a second time.

These goals are discussed in detail below.

In addition, if it is to become successful, a service discovery protocol should be simple enough to implement that virtually any device capable of implementing IP should not have any trouble implementing the service discovery software as well.

4. Service Instance Enumeration

DNS SRV records [RFC 2782] are useful for locating instances of a particular type of service when all the instances are effectively indistinguishable and provide the same service to the client.

For example, SRV records with the (hypothetical) name "_http._tcp.example.com." would allow a client to discover a list of all servers implementing the "_http._tcp" service (i.e. Web servers) for the "example.com." domain. The unstated assumption is that all these servers offer an identical set of Web pages, and it doesn't matter to the client which of the servers it uses, as long as it selects one at random according to the weight and priority rules laid out in RFC 2782.

Instances of other kinds of service are less easily interchangeable. If a word processing application were to look up the (hypothetical) SRV record "_lpr._tcp.example.com." to find the list of printers at Example Co., then picking one at random and printing on it would probably not be what the user wanted.

This proposal borrows the logical service naming syntax and semantics from DNS SRV records, but adds one level of indirection. Instead of requesting records of type "SRV" with name "_lpr._tcp.example.com.", the client requests records of type "PTR" (pointer from one name in the DNS namespace to another). The result of this PTR lookup is a list of zero or more Service Instance Names of the form:

Service Instance Name = <Instance> . <Service> . <Domain>

The <Instance> portion of the name is a single DNS label, containing arbitrary UTF-8-encoded text [RFC 2279]. DNS recommends guidelines for allowable characters for host names [RFC 1034][RFC 1033], but Service Instance Names are not host names. Service Instance Names are not intended to ever be typed in by a normal user; the user selects a Service Instance Name by selecting it from a list of choices presented on the screen. Note that just because this protocol supports arbitrary UTF-8-encoded names doesn't mean that any particular user or administrator setting up a service is obliged to name that service using any characters outside the standard US-ASCII range.

The names resulting from the PTR lookup are presented to the user in a list for the user to select one (or more). Having chosen the desired named instance, the Service Instance Name may then be used immediately, or saved away in some persistent user-preference data structure for future use.

DNS labels are limited to 63 octets in length. UTF-8 encoding can require up to six octets per 31-bit UCS-4 character, which means that in the worst case, the <Instance> portion of a name could be limited to ten characters. However, the UCS-4 characters with longer UTF-8 encodings tend to be the ones which convey greater meaning. A printer name consisting of ten ancient Egyptian Hieroglyphs may well be far more descriptive (to an ancient Egyptian) than a name written in English consisting of just 63 characters.

I welcome input from the IDN Working Group about whether this method of encoding international text is the most appropriate for this particular usage.

There have been proposals to keep the true DNS name of the service typically terse and cryptic, and to use a TXT records attached to that DNS name to hold the 'user-friendly' name which is displayed to the user. The problem with this is that it decouples user perception from reality. Two different instances of services with different DNS names could inadvertently have the same TXT record name, which could be very confusing to users. Maintaining a tight one-to-one mapping between the true DNS name and the 'user-friendly' name as displayed on the screen avoids these anomalies.

There have been questions about why services are not named using Service Instance Names of the form: <Service> . <Instance> . <Domain>

There are three reasons why it is beneficial to name service instances as:

Service Instance Name = <Instance> . <Service> . <Domain>

The first reason is that, the logical decomposition is that a domain has various services; a service has various instances of that service. It does not make sense to say that an instance has various services. These are not host names. The usage model is not, first, what's the name of the host, and then second, what services is it running? The usage model is, first, what's the name of the service, and then second, what are the names of the specific instances of that service?

The second reason is that, when a DNS response contains multiple answers, name compression works more effectively if all the names contain a common suffix. If all the answers in the packet have the same <Service> and <Domain>, then each PTR's rdata only has to give the <Instance> part followed by a two-byte compression pointer.

The third reason is that, this allows subdomains to be delegated along logical service boundaries. For example, the network administrator at Example Co. could choose to delegate the _lpr._tcp.example.com subdomain to a particular machine that has the responsibility to know about all the printers at Example Co. If the service name were the least significant component of the Service Instance Name, then there would be no way to separate the printers from the file servers.

5. Service Name Resolution

Given a particular Service Instance Name, when a client needs to contact that service, it sends a DNS request for the SRV record of that name.

The result of the DNS request is a SRV record giving the port number and target host where the service may be found.

In some environments such as Zeroconf, the host providing the named service may itself not have a well-defined host name. In this case, the 'target' name in the SRV record may simply repeat the same name as the SRV record itself, with an address record attached to the same name giving the appropriate IP address.

In the event that more than one SRV is returned, clients MUST correctly interpret the priority and weight fields -- i.e. Lower numbered priority servers should be used in preference to higher numbered priority servers, and servers with equal priority should be selected randomly in proportion to their relative weights.

Some services discovered via Service Instance Enumeration may need more than just an IP address and port number to properly identify the service. For example, printing via lpr typically specifies a queue name. A file server may have multiple volumes, each identified by its own volume name. A Web server typically has multiple pages, each identified by its own URL. In these cases, the necessary additional data is stored in a TXT record with the same name as the SRV record. The specific nature of that additional data, and how it is to be used, is service-dependent.

6. Selective Queries

This proposal does not attempt to define an arbitrary query language for service discovery, nor do we believe one is necessary.

However, there are some circumstances where narrowing the list of results may be useful. A printing client that wishes to discover only printers that accept Postscript over lpr over TCP should issue a PTR query for the name "_postscript._lpr._tcp.example.com." Only printers that support Postscript should register this PTR record pointing to their name.

Note that the printer's Service Instance Name which this PTR record points to is unchanged -- it is still something of the form "ThePrinter._lpr._tcp.example.com." The domain in which printer SRV records are registered defines the namespace within which printer names are unique. Additional subtypes (e.g. "_postscript") of the basic service type (e.g. "_lpr._tcp") serve to narrow the list of results, not to create more namespace.

The list of possible subtypes, if any, and the additional data stored in TXT records, if any, are defined separately for each basic service type.

7. Populating the DNS with information.

How the SRV and PTR records that describe services and allow them to be enumerated make their way into the DNS is outside the scope of this document. However, it can happen easily in any of a number of ways, for example:

On some networks, the administrator might manually enter the records into the name server's configuration file.

A network monitoring tool could output a standard zone file to be read into a conventional DNS server.

Future IP printers could use Dynamic DNS Update [RFC 2136] to automatically register their SRV and PTR records with the DNS server.

A printer manager device which has knowledge of printers on the network through some other management protocol could also use Dynamic DNS Update [RFC 2136].

Alternatively, a printer manager device could implement enough of the DNS protocol that it is able to answer DNS requests directly, and Example Co.'s main DNS server could delegate the _lpr._tcp.example.com subdomain to the printer manager device.

Zeroconf printers on an unconfigured ad-hoc network answer Multicast DNS requests on their own behalf for appropriate PTR and SRV names within the "local.arpa." domain [mDNS-SC].

8. Relationship to Multicast DNS

This proposal is not strictly related to Multicast DNS, but the two are highly complementary, particularly in Zeroconf environments [ZC].

Lookups for PTR records of the form "<Service>.local.arpa." are defined to use multicast, and return a list of named instances of the form "<Instance>.<Service>.local.arpa."

In Zeroconf environments where state can be transient and configuration information like IP addresses can change at any time, the DNS TTL on SRV and A records should be short, on the order of seconds. However, the DNS TTL on the PTR records pointing to those SRV names should be long, on the order of hours or days, so that once a name has been displayed in some other host's network browser window, the browsing client doesn't have to keep repeatedly asking for the PTR record to make sure it hasn't disappeared.

9. Comparison to Alternative Service Discovery Protocols

At the present time there are many proposed ways to do network service discovery.

The advantage of using DNS is that it makes use of existing software, protocols, infrastructure, and expertise. Existing network analyzer tools already know how to decode and display DNS packets for network debugging.

For ad-hoc networks such as Zeroconf environments, peer-to-peer multicast protocols are appropriate. It is almost certain that the Zeroconf host profile [ZCHP] will specify the use of Multicast DNS for host name resolution in the absence of DNS servers. Given that Zeroconf hosts will have to implement Multicast DNS anyway, it makes sense for them to also perform service discovery using that same Multicast DNS software instead of also having to implement an entirely different service discovery protocol.

In larger networks, a high volume of enterprise-wide IP multicast traffic may not be desirable, so any credible service discovery protocol intended for larger networks has to provide some facility to aggregate registrations and lookups at a central server (or servers) instead of working exclusively using multicast. This requires some service discovery aggregation server software to be written, debugged, deployed, and maintained. This also requires some service discovery registration protocol to be implemented and deployed for clients to register with the central aggregation server. Virtually every company with an IP network already runs DNS server, and DNS already has a dynamic registration protocol [RFC 2136]. Given that virtually every company already has to operate and maintain a DNS server anyway, it makes sense to take advantage of this instead of also having to learn, operate and maintain a different service registration server.

Service discovery needs to be able to provide appropriate security. DNS already has existing mechanisms for security [RFC 2535].

In summary:

> Service discovery requires a central aggregation server.
> DNS already has one: It's called a DNS server.
>
> Service discovery requires a service registration protocol.
> DNS already has one: It's called DNS Dynamic Update.
>
> Service discovery requires a security model.
> DNS already has one: It's called DNSSEC.
>
> Service discovery requires a query protocol
> DNS already has one: It's called DNS.
>
> Service discovery requires a multicast mode for ad-hoc networks.
> DNS doesn't have one right now, but it will soon, to meet Zeroconf requirements.

It makes more sense to use the existing software that every network needs already, instead of deploying an entire parallel system just for service discovery.

10. Real Example

The following examples were prepared using standard unmodified nslookup and standard unmodified BIND running on GNU/Linux. Note: In real life, this information is obtained using graphical network browser software, not command-line tools.

10.1 Question: What printers do we have at example.com?

```
nslookup -q=ptr _lpr._tcp.example.com
_lpr._tcp.example.com   name = Sales._lpr._tcp.example.com
_lpr._tcp.example.com   name = Marketing._lpr._tcp.example.com
_lpr._tcp.example.com   name = Engineering._lpr._tcp.example.com
```

Answer: We have three, called Sales, Marketing, and Engineering.

10.2 Question: What postscript printers do we have at example.com?

```
nslookup -q=ptr _postscript._lpr._tcp.example.com
_postscript._lpr._tcp.example.com  name = Sales._lpr._tcp.example.com
```

Answer: Only Sales is a postscript printer.

10.3 Question: How do I print on Sales?

```
nslookup -q=any Sales._lpr._tcp.example.com
Sales._lpr._tcp.example.com     text = "SPQ"
Sales._lpr._tcp.example.com     priority = 0, weight = 0, port= 49152
        host = bigserver.example.com
bigserver.example.com   internet address = 10.1.2.3
```

Answer: You need to connect to 10.1.2.3, port 49152, queue name "SPQ"

11. IPv6 Considerations

IPv6 has no significant differences, except that the address of the SRV record's target host is given by the appropriate IPv6 address records instead of the IPv4 "A" record.

12. Security Considerations

DNSSEC [RFC 2535] should be used where the authenticity of information is important.

13. IANA Considerations

The IANA will have to allocate symbolic service/protocol names, much as they allocate TCP port numbers today. However, the textual nature of service/protocol names means that there are almost infinitely many more of them available than the finite set of 65535 possible port numbers. It may also be appropriate to allow use of temporary self-assigned service/protocol names, much like the "x-foo/bar" self-assigned experimental MIME types.

14. Copyright

15. References

[mDNS-SC] S. Cheshire, "Performing DNS queries via IP Multicast", Internet-Draft (work in progress), draft-cheshire-dnsext-multicastdns-00.txt, July 2001.

[RFC 2136] P. Vixie, et al., "Dynamic Updates in the Domain Name System (DNS UPDATE)", RFC 2136, April 1997.

[RFC 2279] F. Yergeau, "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

[RFC 2535] D. Eastlake, "Domain Name System Security Extensions", RFC 2535, March 1999.

[RFC 2782] A. Gulbrandsen, et al., "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[ZC]   M. Hattig, "Zeroconf Requirements", Internet-Draft (work in progress), draft-ietf-zeroconf-reqts-08.txt, May 2001.

[ZCHP]   E. Guttman, "Zeroconf Host Profile", Internet-Draft (work in progress), draft-ietf-zeroconf-host-prof-00.txt, July 2001.

16. Author's Address

Stuart Cheshire
Apple Computer, Inc.
1 Infinite Loop
Cupertino
California 95014
USA

Phone: +1 408 974 3207
EMail: rfc@stuartcheshire.org

Stuart Cheshire <cheshire@apple.com>
 * Wizard Without Portfolio, Apple Computer
 * Chairman, IETF ZEROCONF
 * www.stuartcheshire.org

Appendix C
Performing DNS queries via IP Multicast

Stuart Cheshire
Apple Computer
Document: draft-cheshire-dnsext-multicastdns-00.txt
Expires 13th January 2002
13th July 2001

Performing DNS queries via IP Multicast

<draft-cheshire-dnsext-multicastdns-00.txt>

Status of this Memo

Abstract

Multicast DNS is a really obvious idea, whose time has finally come. This draft proposes one possible way of making it work.

1. Acknowledgements

This work builds upon original work done on Multicast DNS by Bill Manning and Bill Woodcock. The authors gratefully acknowledge their contribution to the current specification. Other contributors of valuable ideas include Bernard Aboba, Mark Andrews, Randy Bush, Levon Esibov, James Gilroy, Olafur Gudmundsson, Erik Guttman, Myron Hattig, Thomas Narten, Erik Nordmark and Dave Thaler.

I apologize humbly to anyone who feels their work has not been properly credited and I offer to buy dinner or drinks in compensation.

2. Introduction

This is a rough first draft. Its purpose is to describe the proposed idea well enough for meaningful discussion to take place. As such, while feedback concerning typographical mistakes and similar minutiae is always appreciated, the reader is advised that it is probably unwise to waste a lot of time on such trivia until after we find out whether this proposal will even live long enough to become a 'draft-01'.

When reading this document, familiarity with the concepts of Zero Configuration Networking [ZC] and automatic link-local addressing [v4LL] [RFC 2462] is helpful.

This document proposes no change to the structure of DNS messages, and no new operation codes, response codes, resource record types, or any other new DNS protocol values. This document simply discusses what needs to happen if DNS clients start sending DNS requests to a multicast address.

The primary difference between this document and "draft-ietf-dnsext-mdns-01.txt" is the philosophy about how subdomains of the "local.arpa." domain are delegated. That document proposes that hosts running Multicast DNS Responders each assert an SOA record, thereby claiming to be the sole authority for their own little zone within the "local.arpa." domain. That approach makes it difficult for different hosts to manage two or more resource records with the same name, a feature that has some benefits. This document proposes that subdomains of the "local.arpa." domain can never be delegated, and instead "local.arpa." is managed as a single zone implemented by a loose collection of hosts cooperatively executing a distributed algorithm. From that philosophical difference, a variety of implementation differences emerge.

There has been discussion of whether "local.arpa." is an appropriate domain to use. Perhaps it is not. Perhaps some other domain should, by IETF Standards Action, be declared a reserved name in the DNS protocol for this particular use. In any case, the text "local.arpa." in this document should be taken as a place holder for whatever reserved name or "domain" may eventually be allocated for this purpose.

There has been discussion of how much burden Multicast DNS might impose on a network. It should be remembered that whenever IPv4 hosts communicate they broadcast ARP packets on the network on a regular basis, and this is not disastrous. The approximate amount of multicast traffic generated by hosts using Multicast DNS is anticipated to be roughly the same order of magnitude as the amount of broadcast ARP traffic those hosts already generate.

3. Conventions and Terminology Used in this Document

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in "Key words for use in RFCs to Indicate Requirement Levels" [RFC 2119].

This document uses the term "host name" in the strict sense to mean a fully qualified domain name that has an address record. It does not use the term "host name" in the commonly used but incorrect sense to mean just the first DNS label of a host's fully qualified domain name.

4. Multicast DNS Names

The DNS domain "local.arpa." is (this document proposes) a special domain with special semantics, namely that "local.arpa." and all its subdomains are link-local, and names within this domain are meaningful only on the link where they originate, much as IPv4 addresses in the 169.254/16 prefix are link-local and meaningful only on the link where they originate.

Any DNS query for a name within the "local.arpa." domain MUST be sent to the all-DNS multicast address (224.0.0.251 or its IPv6 equivalent).

It is unimportant whether a name within the "local.arpa." domain occurred because the user explicitly typed in a fully qualified domain name ending in "local.arpa.", or because the user entered an unqualified domain name and the host software appended the "local.arpa." search domain to it. The "local.arpa." domain could appear in the search list because the user manually configured it, or because it was received in a DHCP option, or via any other valid mechanism for configuring the DNS search list. In this respect the "local.arpa." domain is no different to any other search domain that might appear in the list.

DNS queries for a names outside the "local.arpa." domain MAY be sent to the all-DNS multicast address, if no other conventional DNS server is available. This can allow hosts on the same link to continue communicating using each other's globally unique DNS names during network outages which disrupt communication with the greater Internet. This is a contentious issue, and this document does not discuss it in detail, instead concentrating on the issue of resolving local names using DNS packets sent to a multicast address.

A host which belongs to an organization that owns some portion of the DNS namespace can be assigned a globally unique name within that portion of the DNS namespace, for example, "cheshire.apple.com". Another host, attempting and failing to resolve that name via conventional unicast DNS MAY elect to try resolving it via multicast, which may be successful if the two hosts Happen to be on the same link.

However, the majority home customers do not have easy access to any portion of the global DNS namespace within which they have the authority to create names as they wish. This leaves the majority of home computers effectively anonymous for practical purposes. These users MAY elect to give their computers link-local host names of the form: "single-dns-label.local.arpa." For example, my laptop computer answers to the name "stu.local.arpa." Any computer user is granted the authority to name their computer this way, providing that the chosen host name is not already in use on that link. Having named their computer this way, the user has the authority to continue using that name until such time as name conflict occurs on the link which is not resolved in the user's favour. When this happens, the computer (or its human user) SHOULD cease using the name, and may choose to attempt to allocate a new unique name for use on that link.

The point made in the previous paragraph is very important and bears repeating. It is easy for those of us in the IETF community who run our own name servers at home to forget that the majority of computer users do not run their own name server and have no easy way to create their own host names. When these users wish to transfer files between two laptop computers, they are frequently reduced to typing in dotted-decimal IP addresses because they simply have no other way for one host to refer to the other by name. This is a sorry state of affairs.

Allowing ad-hoc allocation of single-label names in a single flat "local.arpa." namespace may seem to invite chaos. However, operational experience with AppleTalk NBP names, which on any given link are also effectively single-label names in a flat namespace, shows that in practice name collisions happen extremely rarely and are not a problem. Groups of computer users from disparate organizations bring Macintosh laptop computers to events such as IETF Meetings, the Mac Hack conference, the Apple World Wide Developer Conference, etc., and complaints at these events about users suffering conflicts and being forced to rename their machines have never been an issue.

Enforcing uniqueness of host names (i.e. the names of DNS address records mapping names to IP addresses) is probably desirable in the common case, but this document does not mandate that. It is also permissible for a collection of coordinated hosts to agree to maintain multiple DNS address records with the same name, possibly for load balancing or fault-tolerance reasons. This document does not take a position on whether that is sensible, but it is important that the Multicast DNS protocol allows hosts to verify and maintain unique names for resource records where that behaviour is desired, and to maintain multiple resource records with a single shared name where that behaviour is desired. This consideration applies to all resource records, not just address records (i.e. host names).

5. IP TTL Checks

A host sending a Multicast DNS request to a link-local address MUST verify that the TTL in reply packets is 255, and silently discard any reply packets where the TTL is not 255. Without this check, it could be possible for remote rogue hosts to send spoof answer packets (perhaps unicast to the victim host) which the receiving machine could misinterpret as having originated on the local link.

There has been some discussion that many current network programming APIs to not provide any indication of the TTL on received packets. This is unfortunate, and should be fixed for hosts that want to be able to guard against spoof packets arriving from off-link.

6. Reverse Address Mapping

Like "local.arpa." the domain "254.169.in-addr.arpa." is defined to be link-local. Any DNS query for a name within the "254.169.in-addr.arpa." domain MUST be sent to the all-DNS multicast address 224.0.0.251.

7. Requesting

There are three kinds of Multicast DNS Requests, one-shot requests of the kind made by today's conventional DNS clients, one-shot requests accumulating multiple replies made by multicast-aware DNS clients, and continuous ongoing Multicast DNS Requests used by IP network browser software.

A Multicast DNS Responder that is offering records that are intended to be unique on the local link MUST also implement a Multicast DNS Requester so that it can first verify the uniqueness of those records before it begins answering requests for them.

7.1 One-Shot Requests

An unsophisticated DNS client may simply send its DNS requests blindly to the 224.0.0.251 multicast address, without necessarily even being aware what a multicast address is. Indeed, certain existing DNS clients (e.g. Mac and Windows) can be persuaded to do this even today, simply by the user typing in that address as the 'name server address'.

Such an unsophisticated DNS client may not get ideal behaviour. Such a client may simply take the first response it receives and fail to wait to see if there are more, but in many instances this may not be a serious problem. If a user types "http://stu.local.arpa." into their Web browser and gets to see the page they were hoping for, then the protocol has met the user's needs in this case.

7.2 One-Shot Requests, Accumulating Multiple Replies

A more sophisticated DNS client should understand that Multicast DNS is not exactly the same as unicast DNS, and should modify its behaviour in some simple ways.

As described above, there are some cases, such as looking up the address associated with a unique host name, where a single response is sufficient, and moreover may be all that is expected. However, there are other DNS requests where more than one response is possible, and for these requests a more sophisticated Multicast DNS client should include the ability to wait for an appropriate period of time to collect multiple responses.

A naive DNS client retransmits its request only so long as it has received no reply. A more sophisticated Multicast DNS client is aware that having received one response is not necessarily an indication that it might not receive others, and has the ability to retransmit its request an appropriate number of times at appropriate intervals until it is satisfied with the collection of responses it has gathered.

A more sophisticated Multicast DNS client that is retransmitting a request for which is has already received some replies, MAY elect to implement duplicate suppression, as described below under "Duplicate Suppression". This indicates to responders who have already replied that their responses have been received, and they don't need to send them again in response to this repeated request.

A Multicast DNS Requester MAY place more than one question into the Question Section of a Multicast DNS Request.

7.3 Continuous Requesting

In One-Shot Requests, with either a single or multiple responses, the underlying assumption is that the transaction begins when the application issues a request, and ends when all the desired responses have been received. There is another type of operation which is more akin to continuous monitoring.

Macintosh users are accustomed to opening the "Chooser" window, selecting a desired printer, and then closing the Chooser window. However, when the desired printer does not appear in the list, the user will typically leave the "Chooser" window open while they go and check to verify that the printer is plugged in, powered on, connected to the Ethernet, etc. While the user jiggles the wires, hits the Ethernet hub, and so forth, they keep an eye on the Chooser window, and when the printer name appears, they know they have fixed whatever the problem was. This can be a useful and intuitive troubleshooting technique, but a user who goes home for the weekend leaving the Chooser window open places a non-trivial burden on the network.

It is important that an IP network browser window displaying live information from the network using Multicast DNS, if left running for an extended period of time, should generate significantly less multicast traffic on the network than the old AppleTalk Chooser.

A Multicast DNS Requester asking the same question repeatedly for an indefinite period of time MUST implement duplicate suppression, as described below.

8. Duplicate Suppression

When a Multicast DNS Requester sends a request to which it already knows some answers, it populates the Answer Section of the DNS message with those cached resource records whose remaining TTL values indicate that they will remain valid for at least the time anticipated to send this DNS request, and the next, and the one after that. For example, if the Multicast DNS Requester is planning to wait four seconds after this request before sending the next, and then eight seconds after that, then only resource records with TTL values greater than twelve seconds should be included in the answer section. This is to ensure that when a resource record's TTL is close to expiration, the Multicast DNS Requester has *two* chances to refresh it before the cached record expires and has to be removed from the list.

A Multicast DNS Responder SHOULD NOT answer a Multicast DNS Request if the answer it would give is already included in the Answer Section with a TTL at least half the correct value. If the TTL of the answer as given in the Answer Section is less than half of the real TTL as known by the Multicast DNS Responder, the responder SHOULD send an answer so as to update the Requester's cache before the record becomes in danger of expiration.

A Multicast DNS Requester MUST NOT cache resource records observed in the Answer Section of other Multicast DNS Requests. The Answer Section of Multicast DNS Requests is not authoritative. By placing information in the Answer Section of a Multicast DNS Request the requester is stating that it *believes* the information to be true. It is not asserting that the information *is* true. Some of those records may have come from other hosts that are no longer on the network. Propagating that stale information to other Multicast DNS Requesters on the network would not be helpful.

A Multicast DNS Responder that implements duplicate suppression SHOULD implement EDNS0 [RFC 2671] to allow larger-sized requests and replies.

9. Responding

A Multicast DNS Responder MUST only reply when it has a positive non-null response to send. Error responses must never be sent. The non-existence of any name in a Multicast DNS Domain is ascertained by the failure of any machine to respond to the Multicast DNS query, not by NXDOMAIN errors.

A Multicast DNS Responder on Ethernet [IEEE802] and similar shared multiple access networks SHOULD delay its responses by a random amount of time selected with uniform random distribution in the range 0-10ms. If multiple Multicast DNS Responders were all to immediately reply to a particular request, a collision would be virtually guaranteed. By imposing a small random delay, the number of collisions is dramatically reduced. 10ms is a short enough time that it is not perceptible to a human user, but long enough to significantly reduce the risk of Ethernet collisions. On a full-sized Ethernet using the maximum cable lengths allowed and the maximum number of repeaters allowed, an Ethernet frame is vulnerable to collisions during the transmission of its first 256 bits. On 10Mb/s Ethernet, this equates to a vulnerable time window of 25.6us.

In the case where a Multicast DNS Responder has good reason to believe that it will be the only responder on the link with a positive non-null response, it MAY reply immediately, without the random delay. To do this safely, it MUST have previously verified that the requested name type and class in the DNS query are unique on this link. This may be appropriate for things like looking up the address record for a particular host name, when the host name has been previously verified unique. This is *not* appropriate for things like looking up PTR records used for DNS Service Discovery [NIAS], where a large number of responses may be anticipated.

Multicast DNS Responses MUST be sent to UDP port 53 (the well-known port assigned to DNS) on the 224.0.0.251 multicast address. Operating in a Zeroconf environment requires constant vigilance. Just because a name has been previously verified unique does not mean it will continue to be so indefinitely. By allowing all Multicast DNS Responders to constantly monitor their peers' responses, conflicts arising out of network topology changes can be promptly detected and resolved.

If the source UDP port in a received Multicast DNS Request is not port 53, this suggests that the client originating the request is an old naive client that is not entirely aware that it is using a multicast address. (The host OS needs to understand what an IP multicast address is in order to hash it to the correct Ethernet multicast address, but the user-level DNS client software does not need to know anything about multicast to blindly send a UDP packet to the IP address 224.0.0.251.) In this case, after sending the usual

Multicast DNS Response to 224.0.0.251 port 53, the Multicast DNS Responder MUST also send a second identical UDP reply to the client via unicast to the request packet's source IP address and port.

Multicast DNS Responders MUST correctly handle DNS request packets containing more than one question, by answering any or all of the questions to which they have answers.

Multicast DNS Responders SHOULD implement EDNS0 [RFC 2671] to allow larger-sized requests and replies. Larger-sized requests are useful to allow longer duplicate suppression lists in the Answer Section.

10. Startup Procedure

Whenever a Multicast DNS Responder starts up, wakes up from sleep, receives an indication of an Ethernet 'Link Change' event, or has any other reason to believe that its network connectivity may have changed in some relevant way, it MUST perform two startup steps.

The first startup step is that for all those resource records that a Multicast DNS Responder desires to be unique on the local link, it MUST send a Multicast DNS Query asking for those resource records, to see if any of them are already in use. The primary example of this is its address record which maps its unique host name to its unique IP address. The ability to place more than one question in a Multicast DNS Request is useful here, because it can allow a host to use a single packet for all of its resource records instead of needing a separate packet for each. If any conflicting Multicast DNS replies are received, then the host MUST defer to the other host already using those names, and MUST select new names for its conflicting records which need to be unique. One second after the first query it should send a second, then two seconds after that a third. If, after a total of seven seconds, no conflicting Multicast DNS replies have been received, the host may move to the second step.

The second startup step is that the Multicast DNS Responder SHOULD send a gratuitous Multicast DNS Response containing, in the Answer Section, all those resource records that may be of interest to other hosts on the link. One example of this is the PTR records used by DNS Service Discovery [NIAS]. Since other hosts running Multicast DNS Requesters may have network browser windows open using an extremely long interval between Multicast DNS Request packets, the reception of a gratuitous Multicast DNS Response from a new device starting up allows the browser window to update immediately instead of having to wait until the next request is sent.

Up to ten of gratuitous Multicast DNS Responses may be sent, providing that the interval between gratuitous responses doubles with every response sent, and the interval between the first two gratuitous responses is not less than one second.

Whenever a Multicast DNS Responder receives any Multicast DNS response (gratuitous or otherwise) containing a conflicting resource record, the conflict MUST be resolved as described below in "Conflict Resolution".

A Multicast DNS Responder MUST NOT send announcements in the absence of information that its network connectivity may have changed in some relevant way. In particular, a Multicast DNS Responder MUST NOT send regular periodic announcements as a matter of course.

11. Conflict Resolution

A conflict occurs when two resource records with the same name, type and class have inconsistent rdata. What may be considered inconsistent is context sensitive, except that resource records with identical rdata are never considered inconsistent, even if they originate from different hosts. In the case of a host desiring to have a unique host name, another address record with the same name but a different IP address is considered inconsistent.

Whenever a Multicast DNS Responder receives any Multicast DNS response (gratuitous or otherwise) containing a conflicting resource record, the Multicast DNS Responder must cease using that record and potentially reconfigure.

In the case of a typical laptop or desktop computer with a human user, reconfiguration is achieved by displaying an error message to the user and suggesting that they choose a new name. In the case of a device with no human operator, reconfiguration is achieved by its software programmatically generating a new name. In either case, the host must then test the new name for uniqueness as described above in "Startup Procedure".

It is important that the host that believes there is a conflict be the one to take action. In the case of two hosts using the same host name, where one has been configured to require a unique host name and the other has not, the one configured to require a unique host name must be the one to reconfigure, since the other one doesn't view the sharing of address records as a conflict and hence sees no reason why it should reconfigure. This algorithm could result in situations where both hosts reconfigure, but this will be rare. The uniqueness check described above in "Startup Procedure" helps reduces resource record conflicts to only those cases where two separate links are connected together, or a previously partitioned link is re-joined.

The examples in this section focus on address records (i.e. host names), but the same considerations apply to all resource records where uniqueness or some other defined constraint is desired.

12. Special Characteristics of Multicast DNS Domains

Unlike conventional DNS, the DNS domains "local.arpa." and "254.169.in-addr.arpa." have only local significance. Conventional DNS seeks to provide a single unified namespace, where a given DNS query yields the same answer no matter where on the planet it is performed or to which recursive DNS server the query is sent. (However, split views, firewalls, intranets and the like have somewhat interfered with this goal of DNS representing a single universal truth). In contrast, each IP link has its own private “local.arpa." and "254.169.in-addr.arpa." namespaces, and the answer to any query for a name within those domains depends on where that query is asked.

Multicast DNS Domains are not delegated from their parent domain via use of NS records. Instead, all Multicast DNS Domains are delegated to the IP address 224.0.0.251 by (potential) IETF Standards Action (i.e. this document, should it become a standard). There are no NS records anywhere in Multicast DNS Domains.

The name server for a Multicast DNS Domain is 224.0.0.251. This is a multicast address; therefore it identifies not a single host but a collection of hosts, working in cooperation to maintain some reasonable facsimile of a competently managed DNS zone. Conceptually a Multicast DNS Domain is a single DNS zone, however its server is implemented as a distributed process running on cluster of loosely cooperating CPUs rather than as a single process running on a single CPU (or tightly coupled multiprocessor).

No delegation is performed within Multicast DNS Domains. Because the cluster of loosely coordinated CPUs is cooperating to administer a single zone, no delegation is necessary or desirable. Just because a particular host on the network may answer queries for a particular record type with the name "example.local.arpa." does not imply anything about whether that host will answer for the name "child.example.local.arpa.", or indeed for other record types with the "example.local.arpa."

Multicast DNS Zones have no SOA record. A conventional DNS zone's SOA record contains information such as the email address of the zone administrator and the monotonically increasing serial number of the last zone modification. There is no single human administrator for any given Multicast DNS Zone, so there is no email address. Because the hosts managing any given Multicast DNS Zone are only loosely coordinated, there is no readily available monotonically increasing serial number to determine whether or not the zone contents have changed. A host holding part of the shared zone could crash or be disconnected from the network at any time without informing the other hosts. There is no reliable way to provide a zone serial number that would, whenever such a crash or disconnection occurred, immediately change to indicate that the contents of the shared zone had changed.

Zone transfers are not possible for any Multicast DNS Zone.

13. Multicast DNS for Service Discovery

This document does not describe using Multicast DNS for network browsing or service discovery. However, the mechanisms this document describes are compatible with (and support) the browsing and service discovery mechanisms proposed in "Discovering Named Instances of Abstract Services using DNS" [NIAS].

This document places few limitations on what DNS record types may be looked up in the "local.arpa." domain. In particular, a Multicast DNS request for the SRV record named "_dns._udp.local.arpa." may yield the port number and host name (and thence IP address) of a conventional DNS server willing to perform general recursive DNS lookups. The benefit of using this mechanism rather than a DHCP option to configure a host's DNS server address is that using DHCP is an outward-looking solution that makes DNS dependent on another protocol, which may not be running on every network (e.g. an IPv6 network using stateless address autoconfiguration [RFC 2462]). Locating a recursive DNS server using Multicast DNS is a self-sufficient solution that reduces DNS's need for support from other protocols. This possibility is not discussed further here.

14. Resource Record TTL Values

Multicast DNS resource records used in typical 'One-Shot' requests should generally have fairly low TTL values, on the order of seconds, rather than hours or days. The transient nature of Zeroconf networks [ZC] [v4LL] means that information can change at any time, and a host caching ancient stale resource records with unreasonably long TTL values could be left trying to work with hopelessly out-of-date information.

Having hosts send gratuitous responses when configuration changes occur can somewhat mitigate this problem, but in the event of a network partition, or temporary signal fade in a wireless network, it is not safe to assume that all hosts will necessarily see all gratuitous responses.

The one exception to this recommendation is resource records expected to be used to populate network browser lists, such as the PTR records used for DNS Service Discovery [NIAS]. Using short TTL values here would force the network browser to be continuously sending Multicast DNS Requests to refresh records before they expired from the list. In this case, the harm done by stale data due to high TTL values is relatively mild. The appearance of names in the network browser list is merely an assertion that the name exists now or has existed in the recent past. In order to actually use any named service, the client has to perform another DNS request to find the IP address, and in the case where the service has been forced to reconfigure to a new IP address (or has left the network entirely), the client will quickly discover that.

15. Enabling and Disabling Multicast DNS

The option to fail-over to Multicast DNS for names outside the "local.arpa." domain SHOULD be a user-configured option, and SHOULD be disabled by default because of the possible security issues related to unintended local resolution of apparently global names.

The option to lookup unqualified (relative) names in the "local.arpa." domain (or not) is controlled by whether or not "local.arpa." appears in the client's DNS search list.

No special control is needed for enabling and disabling Multicast DNS for names within the "local.arpa." domain. The user doesn't need a way to disable Multicast DNS for names within the "local.arpa." domain, because if the user doesn't want to use Multicast DNS, they can achieve this by simply not using names that end in ".local.arpa." If a user *does* enter a name ending in ".local.arpa." into their Web browser, then we can safely assume their intention was probably that it should work. Having user configuration options that can be (intentionally or unintentionally) set so that this doesn't work is just one more way of frustrating the user's ability to perform the tasks they want, perpetuating the view that, "IP networking is too complicated to configure and too hard to use." This in turn perpetuates the continued use of protocols like AppleTalk, and there's no DHCP option to disable that! If we want to retire AppleTalk, we need to offer users equivalent IP functionality that they can rely on to, "always work, like AppleTalk." A little Multicast DNS traffic may be a burden on the network, but it is an insignificant burden compared to continued widespread use of AppleTalk.

16. Considerations for Multiple Interfaces

A host should defend its host name (FQDN) on all active interfaces on which it is answering Multicast DNS requests.

In the event of a name conflict on *any* interface, a host should configure a new host name, if it wishes to maintain uniqueness of its host name.

When answering a Multicast DNS request, a multi-homed host with a link-local address (or addresses) should take care to ensure that any address going out in a Multicast DNS reply is valid for use on the interface on which the reply is going out.

Just as the same link-local IP address may validly be in use simultaneously on different links, the same link-local host name may validly be in use simultaneously on different links, and this is not an error. A multi-homed host with connections to two different links may be able to communicate with two different hosts that are validly using the same name. While this kind of name duplication should be rare, it means that a host which wants to fully support this case needs network programming APIs that allow applications to specify on what interface to perform a link-local Multicast DNS request and/or on what interface a Multicast DNS reply was received.

17. DNS Message Format

This section describes specific restrictions on the allowable values for the header fields of a Multicast DNS message.

17.1. ID (Query Identifier)

Multicast DNS clients SHOULD listen for gratuitous responses issued by hosts booting up (or waking up from sleep or otherwise joining the network). Since these gratuitous responses may contain a useful answer to a question for which the client is currently awaiting an answer, Multicast DNS clients SHOULD examine all received Multicast DNS response messages for useful answers, without regard to the contents of the ID field or the question section. In multicast DNS, knowing which particular query message (if any) is responsible for eliciting a particular response message is less interesting than knowing whether the response message contains useful information.

Multicast DNS clients MAY cache any or all Multicast DNS response messages they receive, for possible future use, providing of course that normal TTL aging is performed on these cashed resource records.

In multicast query messages, the Query ID SHOULD be set to zero on transmission.

In multicast responses, including gratuitous multicast responses, the Query ID MUST be set to zero on transmission, and MUST be ignored on reception.

In unicast response messages generated specifically in response to a particular (unicast or multicast) query, the Query ID MUST match the ID from the query message.

17.2. QR (Query/Response) Bit

In query messages, MUST be zero.

In response messages, MUST be one.

17.3. OPCODE

In both multicast query and multicast response messages, MUST be zero (only standard queries are currently supported over multicast, unless other queries are allowed by future IETF Standards Action).

17.4. AA (Authoritative Answer) Bit

In query messages, the Authoritative Answer bit MUST be zero on transmission, and MUST be ignored on reception.

In response messages for Multicast Domains, the Authoritative Answer bit MUST be one -- not setting this bit implies there's some other place where 'better' information may be found.

17.5. TC (Truncated) Bit

In query messages, the Truncated bit MUST be zero on transmission, and MUST be ignored on reception.

In response messages, if the message does not contain all the data the requester was looking for, the requester SHOULD open a TCP connection to the responder and repeat the query.

17.6. RD (Recursion Desired) Bit

In both multicast query and multicast response messages, the Recursion Desired bit MUST be zero on transmission, and MUST be ignored on reception.

17.7. RA (Recursion Available) Bit

In both multicast query and multicast response messages, the Recursion Available bit MUST be zero on transmission, and MUST be ignored on reception.

17.8. Z (Zero) Bit

In both query and response messages, the Zero bit MUST be zero on transmission, and MUST be ignored on reception.

17.9. AD (Authentic Data) Bit [RFC 2535]

In query messages the Authentic Data bit MUST be zero on transmission, and MUST be ignored on reception.

In response messages, the Authentic Data bit MAY be set. Resolvers receiving response messages with the AD bit set MUST NOT trust the AD bit unless they trust the source of the message and either have a secure path to it or use DNS transaction security.

17.10. CD (Checking Disabled) Bit [RFC 2535]

In query messages, a resolver willing to do cryptography SHOULD set the Checking Disabled bit to permit it to impose its own policies.

In response messages, the Checking Disabled bit MUST be zero on transmission, and MUST be ignored on reception.

17.11. RCODE (Response Code)

In both multicast query and multicast response messages, the Response Code MUST be zero on transmission. Multicast DNS messages received with non-zero Response Codes MUST be silently ignored.

18. IPv6 Considerations

An IPv4-only host and an IPv6-only host behave as "ships that pass in the night". Even if they are on the same Ethernet, neither is aware of the other's traffic. For this reason, each physical link may have *two* unrelated "local.arpa." zones, one for IPv4 and one for IPv6. Since for practical purposes, a group of IPv4-only hosts and a group of IPv6-only hosts on the same Ethernet act as if they were on two entirely separate Ethernet segments, it is unsurprising that their use of the "local.arpa." zone should occur exactly as it would if they really were on two entirely separate Ethernet segments.

A dual-stack (v4/v6) host can participate in both "local.arpa." zones, and should register its name(s) and perform its lookups both using IPv4 and IPv6. This enables it to reach, and be reached by, both IPv4-only and IPv6-only hosts.

There has been discussion of the proposal that in the IPv6 case, the all-DNS multicast address should not be a single address, but instead a range of addresses selected using a hash function of the name being looked for. There are some issues with this:

1. The hash function must work correctly with both normal (case-insensitive) DNS labels and binary labels [RFC 2673].

2. This may prevent more than one question being put into a single packet, since the different questions may hash to different multicast addresses.

3. This impedes the ability to use a single multicast reply packet to answer the client and simultaneously facilitate ongoing conflict monitoring, because every client would have to listen on every multicast address in the range (or rapidly join and leave multicast groups on demand for each request) in order to receive the reply.

4. This limits the ability to gain certain useful functionality out of old resolver software by configuring it with a single All-DNS multicast address to which it can send its queries.

19. Security Considerations

DNSSEC [RFC 2535] should be used where the authenticity of information is important.

When DNS queries for names outside the "local.arpa." domain are sent to the all-DNS multicast address (during of network outages which disrupt communication with the greater Internet) it is *especially* important to use DNSSEC, because the user may have the impression that he or she is communicating with some authentic host, when in fact he or she is really communicating with some local host that is merely masquerading as that name. This is less critical for names within the "local.arpa." domain, because within this domain the user can be aware that names have only local significance and no global authority is implied.

Most computer users neglect to type the trailing dot at the end of a fully qualified domain name, making it a relative domain name (e.g. "www.example.com"). In the event of network outage, attempts to positively resolve the name as entered will fail, resulting in application of the search list, including "local.arpa.", if present. A malicious host could masquerade as "www.example.com" by answering the resulting Multicast DNS request for "www.example.com.local.arpa". To avoid this, a host MUST NOT append the search domain "local.arpa.", if present, to any relative (partially qualified) domain name containing two or more labels. Appending "local.arpa." to single-label relative domain names is acceptable, since the user should have no expectation that a single-label domain name will resolve as-is.

[Lots more work to be done here!]

20. IANA Considerations

The IANA has allocated the IPv4 link-local multicast address 224.0.0.251 for the use described in this document.

We'd like the IANA to designate the DNS domain "local.arpa." a "Multicast Domain" with special semantics, namely that "local.arpa." and its subdomains are link-local, and names within this domain are meaningful only on the link where they originate, much as IPv4 addresses in the 169.254/16 prefix are link-local and meaningful only on the link where they originate. Likewise we'd like the IANA to designate the DNS domain "254.169.in-addr.arpa." to be similarly link-local and non-delegated.

No other IANA services are required by this document.

21. Copyright

22. References

[IEEE802] IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture. Institute of Electrical and Electronic Engineers, IEEE Standard 802, 1990.

[NIAS] S. Cheshire, "Discovering Named Instances of Abstract Services using DNS", Internet-Draft (work in progress), draft-cheshire-dnsext-nias-00.txt, July 2001.

[RFC 2119] S. Bradner, "Key words for use in RFCs to Indicate Requirement Levels", RFC 2119, March 1997.

[RFC 2462] S. Thomson and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998.

[RFC 2535] D. Eastlake, "Domain Name System Security Extensions", RFC 2535, March 1999.

[RFC 2671] P. Vixie, "Extension mechanisms for DNS (EDNS0)", RFC 2671, August 1999.

[RFC 2673] M. Crawford, "Binary Labels in the Domain Name System", RFC 2673, August 1999.

[v4LL] S. Cheshire and B. Aboba, "Dynamic Configuration of IPv4 Link-Local Addresses", Internet-Draft (work in progress), draft-ietf-zeroconf-ipv4-linklocal-03.txt, June 2001.

[ZC] M. Hattig, "Zeroconf Requirements", Internet-Draft (work in progress), draft-ietf-zeroconf-reqts-08.txt, May 2001.

23. Author's Address

Stuart Cheshire
Apple Computer, Inc.
1 Infinite Loop
Cupertino
California 95014
USA

Phone: +1 408 974 3207
EMail: rfc@stuartcheshire.org

Appendix D
The DISCOVER opcode

Individual Submission
draft-ymbk-opcode-discover-03.txt

Bill Manning, ISI
Paul Vixie, ISC
Erik Guttman, SUN

25 Oct 2001

The DISCOVER opcode

The capitalized keywords "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119

Abstract:

The QUERY opcode in the DNS is designed for unicast. With the development of multicast capabilities in the DNS, it is desireable to have a more robust opcode for server interactions since a single request may result in replies from multiple responders. So DISCOVER is defined to deal with replies from multiple responders.

As such, this document extend the core DNS specifications to allow clients to have a method for coping with replies from multiple responders. Use of this new opcode may facilitate DNS operations in modern networking topologies. A prototype of the DISCOVER opcode was developed as part of the TBDS project, funded under DARPA grant F30602-99-1-0523.

Introduction:

This document describes an experimental extension to the DNS to receive multiple responses which is the likely result when using DNS that has enabled multicast queries. This approach was developed as part of the TBDS research project, funded under DARPA grant F30602-99-1-0523. The full processing rules are documented here for possible incorporation in a future revision of the DNS specification."

Method:

DISCOVER works like QUERY except:

1. It can be sent to a broadcast or multicast destination (QUERY isn't defined for non-unicast, and arguably shouldn't be.) While DISCOVER could be used for unicast, what is the point?

2. The Question section, if present, has <QNAME=zonename,QTYPE=SOA> tuples. Future work could augment this structure as follows: <QNAME=service,QTYPE=SRV>

3. If QDCOUNT==0 then only servers willing to do recursion should answer. Other servers must silently discard the DISCOVER request.

4. If QDCOUNT!=0 then only servers who are authoritative for the zones named by some QNAME should answer.

5. Responses may echo the request's Question section or leave it blank.

6. Responses have "normal" Answer, Authority, and Additional sections. e.g. the response is the same as that to a QUERY. It is desireable that zero content answers not be sent to avoid badly formed or unfulfilled requests. Responses should be sent to the unicast address of the requester and the source address should reflect the unicast address of the responder.

Example usage for gethostby{name,addr}-style requestors:

Compute the zone name of the enclosing in-addr.arpa or ip6.int domain.

DISCOVER whether anyone in-scope is authoritative for this zone.

If so, query these authoritative servers for local in-addr/ip6 names.

If not, DISCOVER whether there are recursive servers available.

If so, query these recursive servers for local in-addr/ip6 names.

So, a node will issue a multicast request with the DISCOVER opcode at some particular multicast scope. Then determine, from the replies, whether there are any DNS servers which are authoritative (or support recursion) for the zone. Replies to DISCOVER requests MUST set the Recursion Available (RA) flag in the DNS message header.

It is important to recognize that a requester must be prepared to receive multiple replies from multiple responders.

Once one learns a host's FQDN by the above means, repeat the process for discovering the closest enclosing authoritative server of such local name.

Cache all NS and A data learned in this process, respecting TTL's.

Usage for SRV requestors:

Do the gethostbyaddr() and gethostbyname() on one's own link-local address, using the above process.

Assume that the closest enclosing zone for which an authority server answers an in-scope DISCOVER packet is "this host's parent domain".

Compute the SRV name as _service._transport.*.parentdomain.

This is a change to the definition as defined in RFC 1034. A wildcard label ("*") in the QNAME used in a DNS message with opcode DISCOVER SHOULD be evaluated with special rules. The wildcard matches any label for which the DNS server data is authoritative. For example 'x.*.example.com.' would match 'x.y.example.com.' and 'x.yy.example.com.' provided that the server was authoritative for 'example.com.' In this particular case, we suggest the follwing considerations be made:

getservbyname() can be satisfied by issuing a request with this computed SRV name. The servent structure can be populated by values returned from a request as follows:

s_name	The name of the service, "_service" without the preceding underscore.
s_aliases	The names returned in the SRV RRs in replies to the query.
s_port	The port number in the SRV RRs replies to the query. If these port numbers disagree - one of the port numbers is chosen, and only those names which correspond are returned.
s_proto	The transport protocol from named by the "_transport" label, without the preceding underscore.

Send SRV query for this name to discovered local authority servers.

Usage for disconnected networks with no authority servers:

Hosts should run a "stub server" which acts as though its FQDN is a zone name. Computed SOA gives the host's FQDN as MNAME, "." as the ANAME, seconds-since-1Jan2000 as the SERIAL, low constants for EXPIRE and the other timers. Computed NS gives the host's FQDN. Computed glue gives the host's link-local address. Or Hosts may run a "DNS stub server" which acts as though its FQDN is a zone name. The rules governing the behavior of this stub server are given elsewhere [1] [2].

Such stub servers should answer DISCOVER packets for its zone, and will be found by the iterative "discover closest enclosing authority server" by DISCOVER clients, either in the gethostbyname() or SRV cases described above. Note that stub servers only answer with zone names which match QNAME's, not with zone names which are owned by QNAME's.

The only deviation from the DNS[3][4] model is that a host (like, say, a printer offering LPD services) has a DNS server which answers authoritatively for something which hasn't been delegated to it. However, the only way that such DNS servers can be discovered is with a new opcode, DISCOVER, which is explicitly defined to discover undelegated zones for tightly scoped purposes. Therefore this isn't officially a violation of DNS's coherency principles.

IANA Considerations

As a new opcode, the IANA will need to assign a numeric value for the memnonic. The last OPCODE assigned was "5", for UPDATE. Test implementations have used OPCODE "6".

Security Considerations

No new security considerations are known to be introduced with a new opcode, however using multicast for service discovery has the potential for denial of service, primarily from flooding attacks. It may also be possible to enable deliberate misconfiguration of clients simply by running a malicious DNS resolver that claims to be authoritative for things that it is not. One possible way to mitigate this effect is by use of credentials, such as CERT resource records within an RR set. The TBDS project took this approach.

5. Attribution:

This material was generated in discussions on the mdns mailing list hosted by Zocalo in March 2000. Paul Vixie, Stuart Cheshire, Bill Woodcock, Erik Guttman and Bill Manning were active contributors.

6. Author's Address

Bill Manning
PO 12317
Marina del Rey, CA. 90295
+1.310.322.8102
bmanning@karoshi.com

Paul Vixie
Internet Software Consortium
950 Charter Street
Redwood City, CA 94063
+1 650 779 7001
<vixie@isc.org>

Erik Guttman
Sun Microsystems
Eichhòlzelstr. 7
74915 Waibstadt Germany
+49 6227 356 202
erik.guttman@sun.com

7. References

[1] draft-ietf-dnsext-mdns-00.txt
[2] draft-manning-dnsext-mdns-00.txt
[3] RFC 1034
[4] RFC 1035

--bill